ELSEVIER

Discrete Applied Mathematics 78 (1997) 207–226

DISCRETE APPLIED MATHEMATICS

The point-to-point connection problem – analysis and algorithms

Madan Natu, Shu-Cherng Fang*

Operations Research and Industrial Engineering, P.O. Box 7913, North Carolina State University, Raleigh, NC 27695-7913, USA

Received 23 December 1994; revised 25 July 1996; accepted 2 December 1996

Abstract

The point-to-point connection problem is to find a subset of arcs with minimal total cost connecting a fixed number of source–destination pairs. The problem has many variations for different applications. In this paper, we focus on a case where the source–destination pairs are prematched. We examine the structure of the problem with two source–destination pairs and provide an efficient implementation of a Dijkstra-like algorithm with time complexity $\mathcal{O}(mn + n^2 \log n)$. We also provide a dynamic programming algorithm with complexity $\mathcal{O}(n^{1.5})$ for the case with three source–destination pairs. We conjecture that the same approach can be generalized for p source–destination pairs with complexity $\mathcal{O}(n^{3p+2})$ where p is fixed.

Keywords: Algorithms; Point-to-point connection problem; Dynamic programming; Computational complexity

1. Introduction

Let $G = (V, E)$ be a directed network with sources $s_1, s_2, \ldots, s_p \in V$, corresponding destinations $t_1, t_2, \ldots, t_p \in V$, and a positive cost $c_{i,j}$ for each arc $i \to j \in E$. The point-to-point connection problem is to find a subset $E' \subseteq E$ in G such that each source is connected to its corresponding destination by a path and the total cost $\sum_{i \to j \in E'} c_{i,j}$ is minimized.

The point-to-point connection problem is equivalent to finding a minimal cost network while maintaining connection between sources and destinations. The problem has applications in the design and layout phase of communication and transportation networks where the total cost corresponds to the fixed set-up cost of the

* Corresponding author. Tel.: (919) 515-2350; fax: (919) 515-5281; e-mail: fang@eos.ncsu.edu.

0166-218X/97/$17.00
PII S0166-218X(97)00010-3

network. Our model is a very simplified version of this problem. Also, it can be regarded as a special case of the *multiconnected Steiner network problem* (see [3, 13]), or of the *network design problem* (see [11]).

This problem is closely related to the Steiner tree problem and the point-to-point delivery problem. Hwang et al. [8] provided a state of the art study on the Steiner tree problem. Dreyfus and Wagner [4] gave an exact algorithm for the Steiner tree problem on undirected graphs. The case with a single source and p destinations in directed networks is equivalent to the minimum weight p-terminal Steiner arborescence problem. An excellent source for this problem is Hwang et al. [8]. The two-terminal Steiner arborescence problem has been studied before by Ball et al. [1]. They have discussed different linear programming formulations of the problem. A problem in directed networks, closely related to the Steiner Arborescence problem, is the Steiner equivalent subgraph problem. Hakimi [7] had discussed this problem. Also, Martello and Toth [12] provided a branch-and-bound algorithm. The special case of series-parallel networks was discussed by Ritchey et al. [15]. Applications of the point-to-point delivery problem in routing, package delivery and rail freight shipping were discussed in detail with heuristics by Leung et al. [9].

Recently, Li et al. [10] discussed the computational complexity of both the point-to-point delivery and connection problems. They considered different variations of these problems and proved that most cases are $\mathcal{NP}$-hard. A general approximation technique proposed by Goemans and Williamson [6] can be applied to the point-to-point connection problem on undirected graphs (where p is part of the input). They guarantee a constant worst case bound of $(2 - 1/p)$ on the performance of the heuristic. Li et al. [10] also gave a dynamic programming algorithm for the point-to-point connection problem with two source–destination pairs and left the case with three or more source–destination pairs as an interesting open problem. However, their dynamic programming formulation does not provide enough insights. Moreover, their algorithm is of $\mathcal{O}(n^5)$ complexity. Natu and Fang [14] recently proposed a more elegant approach with complexity $\mathcal{O}(n^4)$. Here, we further provide an efficient implementation of a Dijkstra-like algorithm with improved complexity of $\mathcal{O}(mn + n^2 \log n)$. The point-to-point connection problem with three source -destination pairs is also analysed in this paper by using different possible configurations of an optimal solution. A dynamic programming algorithm with complexity $\mathcal{O}(n^{11})$ is provided. The analysis helps us understand the decomposition used in the dynamic programming algorithm and indicates why it is difficult to generalize this constructive proof for $p > 3$. However, we conjecture that the same approach can be generalized for $p > 3$ with complexity $\mathcal{O}(n^{3p+2})$.

In Section 2, we provide insights into the structure of an optimal solution for $p = 2$, and then exploit this special structure to provide a Dijkstra-like algorithm. In Section 3, a dynamic programming algorithm is given for $p = 3$. Section 4 deals with the conjecture for $p > 3$ and discusses the difficulty in generalizing the proof. Concluding remarks are made in Section 5.

2. The point-to-point connection problem with $p=2$

2.1. The structure of an optimal solution

Given two source–destination pairs (s_1, t_1) and (s_2, t_2), let c_{uv} be the cost of the arc between nodes u and v, $C(u, v)$ the cost of any arbitrary path from u to v, and $C^*(u, v)$ the cost of the shortest path (i.e., minimum cost path) from node u to node v. Let us define $C^*(u, u) = 0$, for all $u \in V$, and $C^*(u, v) = +\infty$, if no path exists between u and v. Also, let us denote a path between s_1, t_1 by P_1 and a path between s_2, t_2 by P_2. We define P as a *shared segment* of paths P_1 and P_2, if P is a maximal common subpath of P_1 and P_2 where "maximal" is defined with respect to the number of arcs in the common subpath. Two nodes u and v are said to be (u, v)-*connected* if there exists a directed path from u to v and is denoted by $u \rightsquigarrow v$.

Fig. 1 shows that the optimal solution cannot have the structure with one subpath going along $v_1^1 \rightsquigarrow i \rightsquigarrow v_2^2$ and other subpath going along $v_1^1 \rightsquigarrow j \rightsquigarrow v_2^2$. Even if those two subpaths have the same total cost, the optimal solution can still be rearranged to include just one subpath. Please note that for all figures in this section a directed arrow between any two nodes of the network actually represents a directed *path* between those nodes.

It is also clear that if an optimal solution consisting of two paths P_1^* and P_2^* has k shared segments, then they must be shared in the reverse order as illustrated in Fig. 2. In this figure, path P_1^* from s_1 to t_1 passes through segments $v_1^1 \rightsquigarrow v_2^1, v_1^2 \rightsquigarrow v_2^2, \ldots, v_1^k \rightsquigarrow v_2^k$ in that order, while P_2^* shares them with P_1^*, but exactly in reverse order.

Fig. 3 is only a rearrangement of Fig. 2. Clearly, the pair of paths can be constructed by adding subpaths $s_1 \rightsquigarrow v_1^1 \rightsquigarrow v_2^1 \rightsquigarrow t_2$, $v_2^1 \rightsquigarrow v_1^2 \rightsquigarrow v_2^2 \rightsquigarrow v_1^1, \ldots, v_2^{k-1} \rightsquigarrow v_1^k \rightsquigarrow v_2^k \rightsquigarrow v_1^{k-1}$ like a ladder and then finally adding subpaths $s_2 \rightsquigarrow v_1^k$, and $v_2^k \rightsquigarrow t_1$. We call this the *ladder property*.

Fig. 4 illustrates the structure of an optimal solution for a simple example. The cost of the optimal solution is 26.

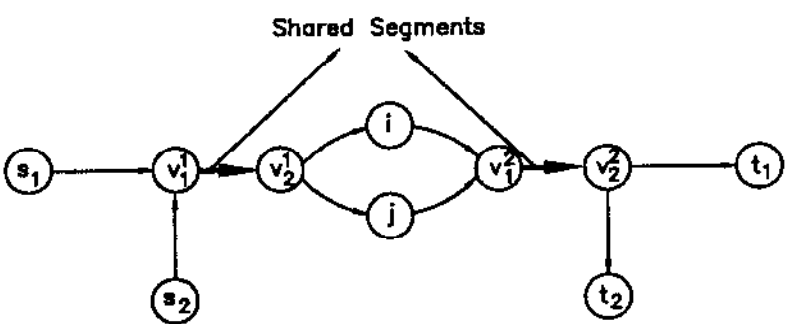


Fig. 1. A feasible but non-optimal solution to the connection problem.

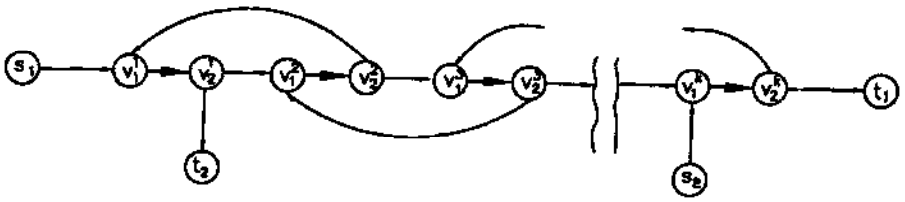


Fig. 2. The property of an optimal solution.

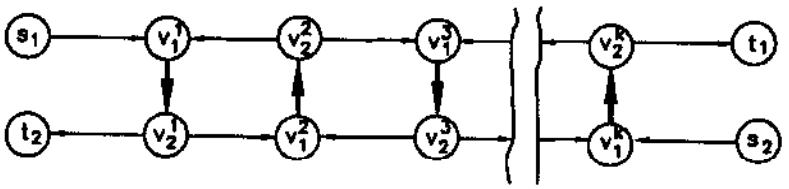


Fig. 3. A rearrangement of an optimal solution as a ladder.

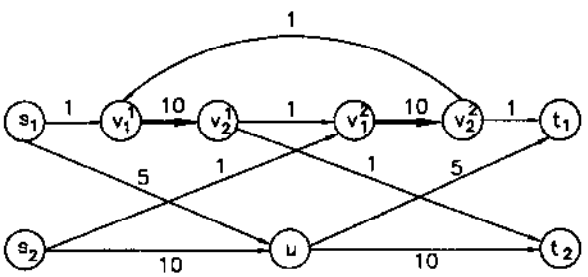


Fig. 4. An illustration of time constraint.

In the next section, the ladder property of an optimal solution will enable us to decompose the problem into several subproblems and to transform the problem into a shortest path problem with $\mathcal{O}(n^2)$ nodes, where n is the number of nodes of the original network. We will refer to the arcs between s_1 and v_1^1, arcs between v_2^1 and v_1^2, ..., and arcs between v_2^k and t_1 as forward progress arcs for P_1^*; similarly, the arcs between v_1^1 and t_2, arcs between v_2^2 and v_1^1, ..., and arcs between s_2 and v_1^k will be referred to as backward progress arcs for P_2^*. Also, we will refer to the shared segments as reversal arcs, because, they are used by P_1^* and P_2^* in reverse order. The ladder property will also be useful in the design of an algorithm for the case with $p = 3$.

2.2. A Dijkstra-like algorithm

As mentioned in the last section, Li et al. [10] have given an algorithm for the point-to-point connection problem with two source–destination pairs. But, their

algorithm is of $\mathcal{O}(n^5)$ complexity. Natu and Fang [14] recently proposed an approach with complexity $\mathcal{O}(n^4)$. First, we show the construction in that approach. Then, we provide an algorithm that resembles Dijkstra's shortest path algorithm. This algorithm has an improved complexity of $\mathcal{O}(mn + n^2 \log n)$.

We use the original directed network to construct a "2-dimensional" network $\bar{G} = (\bar{V}, \bar{E} \equiv \bar{E}^1 \cup \bar{E}^2)$ where

- $\bar{V} = \{\langle v_1, v_2\rangle \mid v_1, v_2 \in V\} = V \times V$,
- $\bar{E}^1 = \{\langle v_1, v_2\rangle \to \langle v_1', v_2'\rangle \mid v_2 \rightsquigarrow v_1' \rightsquigarrow v_2' \rightsquigarrow v_1$, and $\langle v_1, v_2\rangle \neq \langle s_2, t_1\rangle, \langle v_1, v_2\rangle \neq \langle v_1', v_2'\rangle\}$,
- $\bar{E}^2 = \{\langle v_1, v_2\rangle \to \langle s_2, t_1\rangle \mid s_2 \rightsquigarrow v_1, v_2 \rightsquigarrow t_1, \langle v_1, v_2\rangle \neq \langle s_2, t_1\rangle\}$.

Also, in $\bar{E}^1$, let the cost of $\langle v_1, v_2\rangle \to \langle v_1', v_2'\rangle$ be $C^*(v_2, v_1') + C^*(v_1', v_2') + C^*(v_2', v_1)$ and in $\bar{E}^2$, let the cost of $\langle v_1, v_2\rangle \to \langle s_2, t_1\rangle$ be $C^*(s_2, v_1) + C^*(v_2, t_1)$.

We can prove that an optimal solution to the *point-to-point connection problem* with two source–destination pairs can be obtained by solving a shortest path problem on the new network. However, here we discuss a different construction which ensures an efficient algorithm.

Again, we use the original directed network to construct a "2-dimensional" network $G = (V, E = E^1 \cup E^2)$ where

- $\bar{V} = \{\langle v_1, v_2\rangle \mid v_1, v_2 \in V\} = V \times V$,
- $\bar{E}^1 = \{\langle v_1, v_2\rangle \to \langle v_1', v_2\rangle \mid v_1 \to v_1' \in E, v_2 \in V\}$
  $\cup \{\langle v_1, v_2'\rangle \to \langle v_1, v_2\rangle \mid v_2 \to v_2' \in E, v_1 \in V\}$,
- $\bar{E}^2 = \{\langle v_1, v_2\rangle \to \langle v_2, v_1\rangle \mid \text{for } v_1, v_2 \in V\}$.

The construction defines a function $f : S(G) \mapsto S(\bar{G})$, where $S(G)$ is the set of all directed networks and $S(\bar{G})$ is the set which contains all possible $\bar{G}$. The arc-sets $\bar{E}^1$ and $\bar{E}^2$ correspond to the set of progress arcs and the set of reversal arcs respectively. Also, in $\bar{E}^1$, the cost of an arc is assigned to be the cost of the corresponding arc (i.e., $c_{v_1 v_1'}$ or $c_{v_2 v_2'}$) in the original network. In $\bar{E}^2$, the cost of an arc is the cost of the shortest path from v_1 to v_2, i.e., $C^*(v_1, v_2)$.

Consider the set of pairs of walks between the two source–destination pairs. (Please see [2] for the definition of a walk). It is sufficient to consider the pairs of walks that satisfy the ladder property of Section 2.1. We denote this set by S_G. Let $S_{\bar{G}}$ be the set of paths from $\langle s_1, t_2\rangle$ to $\langle t_1, s_2\rangle$ in $\bar{G}$. Then, we have the following two lemmas.

Lemma 1. *For every path P in $\bar{G}$, there exists a pair of walks in G with the same cost.*

Proof. Let

$$P \equiv \langle s_1, t_2\rangle \rightsquigarrow \langle v_1^1, v_2^1\rangle \to \langle v_2^1, v_1^1\rangle \rightsquigarrow \langle v_1^2, v_2^2\rangle \to \langle v_2^2, v_1^2\rangle \rightsquigarrow \cdots \rightsquigarrow \langle v_1^k, v_2^k\rangle \to \langle v_2^k, v_1^k\rangle \rightsquigarrow \langle t_1, s_2\rangle$$

be a directed path in $\bar{G}$. Noting that only the reversal arcs have been indicated here, from the construction of $\bar{G}$, we can find directed walks P_1 and P_2 in G such that

$$P_1 \equiv s_1 \rightsquigarrow v_1^1 \rightsquigarrow v_2^1 \rightsquigarrow v_1^2 \rightsquigarrow \cdots \rightsquigarrow v_2^k \rightsquigarrow t_1,$$

$$P_2 \equiv s_2 \rightsquigarrow v_1^k \rightsquigarrow v_2^k \rightsquigarrow v_1^{k-1} \rightsquigarrow \cdots \rightsquigarrow v_2^1 \rightsquigarrow t_2.$$

The intermediate nodes on the unshared segments of the directed walks P_1 and P_2 are obtained from the intermediate nodes on the directed path P. Also, the intermediate nodes on the shared segments are obtained from the shortest paths between their end nodes. This shows that for any directed path P in $\bar{G}$, there exists a pair of walks $(P_1, P_2) \in G$ with the ladder property. Hence the proof is complete. □

Lemma 2. *For every pair of walks (P_1, P_2) in G, there exists a path P in $\bar{G}$ such that*

$$\sum_{u \to v \in E_{(P_1,P_2)}} c_{uv} \geqslant \sum_{u \to v \in E_P} c_{uv}$$

where $E_{(P_1,P_2)}$ denotes the set of arcs belonging to the pair of walks (P_1, P_2), and E_P is the set of arcs on the path P.

Proof. We have to prove that for any given element $(P_1, P_2) \in S_G$ with k shared segments, there exists a directed path $P \in S_{\bar{G}}$ that satisfies the above cost inequality.

Let $v_1^1, v_2^1, v_1^2, v_2^2, \ldots, v_1^k, v_2^k$ be the start and end nodes of the shared segments between the walks P_1 and P_2. Define

$$P \equiv \langle s_1, t_2 \rangle \rightsquigarrow \langle v_1^1, v_2^1 \rangle \to \langle v_2^1, v_1^1 \rangle \rightsquigarrow \langle v_1^2, v_2^2 \rangle \to \langle v_2^2, v_1^2 \rangle \rightsquigarrow \cdots \rightsquigarrow \langle v_1^k, v_2^k \rangle \to \langle v_2^k, v_1^k \rangle \rightsquigarrow \langle t_1, s_2 \rangle.$$

The intermediate arcs in P as defined above are obtained from the corresponding unshared arcs on the walks, P_1 and P_2. In this way, $P \in S_{\bar{G}}$ and the total cost of P is

$$C_1 = \sum_{u \to v \in E_P} c_{uv}$$

This defines a function f_2.

Let C_2 be the cost of $(P_1, P_2) \in S_G$. Then

$$\begin{aligned} C_2 &= \{C(s_1, v_1^1) + C(v_1^1, v_2^1) + C(v_2^1, t_2)\} + \{C(v_2^1, v_1^2) + C(v_1^2, v_2^2) + C(v_2^2, v_1^1)\} \\ &\quad + \cdots + \{C(v_2^{k-1}, v_1^k) + C(v_1^k, v_2^k) + C(v_2^k, v_1^{k-1})\} + \{C(v_2^k, t_1) + C(s_2, v_1^k)\} \\ &\geqslant \{C(s_1, v_1^1) + C^*(v_1^1, v_2^1) + C(v_2^1, t_2)\} + \{C(v_2^1, v_1^2) + C^*(v_1^2, v_2^2) + C(v_2^2, v_1^1)\} \\ &\quad + \cdots + \{C(v_2^{k-1}, v_1^k) + C(v_1^k, v_2^k) + C(v_2^k, v_1^{k-1})\} + \{C(v_2^k, t_1) + C(s_2, v_1^k)\} \\ &= C_1. \end{aligned}$$

Since we have replaced every shared segment with a corresponding shared segment of the shortest path, the cost inequality follows. □

Theorem 3. *The point-to-point connection problem on a directed network with two source–destination pairs can be solved in* $\mathcal{O}(mn + n^2 \log n)$ *time.*

Proof. The set of pairs of paths in network G with the ladder property is a subset of S_G. Hence, as a direct consequence of Lemmas 1 and 2, the connection problem on network G is equivalent to a shortest path problem on $\bar{G}$.

The optimal solution of the point-to-point connection problem belongs to the set S_G. Lemma 2 partitions the set S_G in several subsets. All elements in each subset map with a single element of $S_{\bar{G}}$ and it satisfies the cost inequality. Thus, the cost of a least-cost element of $S_{\bar{G}}$ is less than the cost of any element of S_G. Also, Lemma 1 guarantees the existence of an element in S_G with the same cost. Hence, the point-to-point connection problem with two source–destination pairs is equivalent to a shortest path problem on the new network $\bar{G}$.

Since the network $\bar{G}$ has (n^2) nodes and $(2mn + n^2)$ arcs, the construction of the new network takes $\mathcal{O}(mn)$ time. To speed up the computation, we can preprocess to get $C^*(u, v)$ in $\mathcal{O}(mn + n^2 \log n)$ time, using Fibonacci heaps, as suggested by Fredman and Tarjan [5] for finding all pairs shortest paths. The values of these shortest paths represent the costs of all reversal arcs in the new network $\bar{G}$. Dijkstra's shortest path algorithm takes $\mathcal{O}(mn + n^2 \log n)$ time on the network $\bar{G}$, and hence the overall complexity of our algorithm is $\mathcal{O}(mn + n^2 \log n)$. □

3. The point-to-point connection problem with $p=3$

In Section 2.1, we examined the structure of an optimal solution for $p = 2$ and made use of this to devise an efficient algorithm in Section 2.2. This simple structure is lost for $p \geqslant 3$. In this section, we examine the properties of an optimal solution when $p = 3$. In Section 4, we will see that such an analysis becomes rather difficult when $p > 3$, because of the highly complex structure of an optimal solution.

The definitions and notations introduced in Section 2.1 remain valid except that some definitions must be modified and some notations should be appropriately extended. Let us denote a path between source s_i and destination t_i by P_i. From now on, a shared segment refers to a maximal common subpath between any paths linking the corresponding p ($\geqslant 3$) source–destination pairs. It is maximal with respect to two properties: (i) it is maximal in the number of source–destination pairs which share the same segment, and (ii) it is not contained in any subpath with more arcs and shared by the same source–destination paths. In other words, any arc of this maximal common subpath must not be used by any other source–destination connecting path. Figs. 5 and 6 are used to clarify this definition. In Fig. 5 there are three maximal shared segments, namely, $v_1^1 \rightsquigarrow v_2^1$, shared by paths P_1 and P_2; $v_2^1 \rightsquigarrow v_1^2$, shared by paths P_1, P_2 and P_3; and $v_1^2 \rightsquigarrow v_2^2$, again shared by paths P_1 and P_2. Similarly, we have 3 shared segments in Fig. 6.

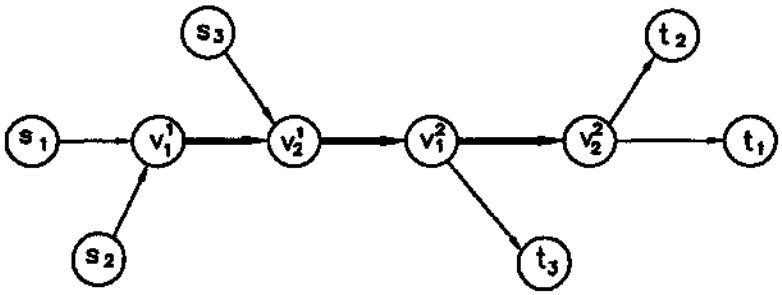


Fig. 5. An illustration of the definition of a shared segment: Case 1.

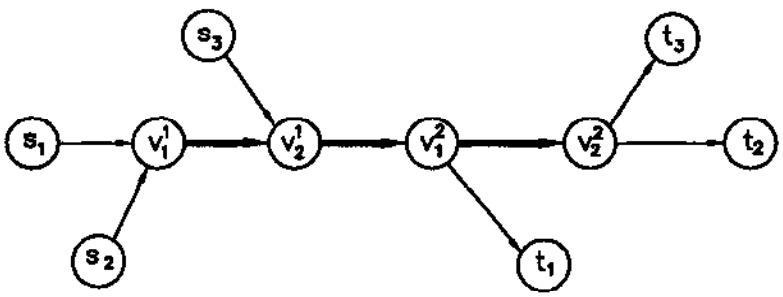


Fig. 6. An illustration of the definition of a shared segment: Case 2.

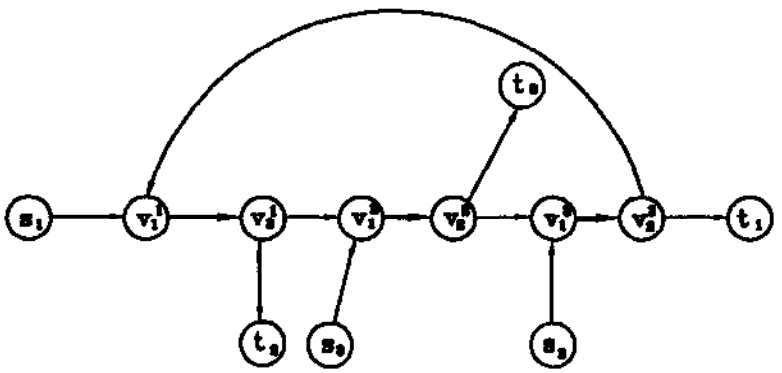
Note that, in our study, since there is no capacity on any arc of a given network, any number of paths can actually *use* the same arc without *sharing* them. But, while calculating the cost of the solution in such a case, the cost of that arc must be accounted for each path *using* it. This observation is important for the dynamic programming algorithm discussed later.

For $p = 3$, the structure of an optimal solution is not unique and hence, we have to consider all *symmetric* and *degenerate* configurations in the analysis. A *symmetric* configuration is obtained from one configuration by cyclically permuting the source–destination nodes, while a *degenerate* configuration is obtained by changing the number of shared segments between two source–destination pairs to 0 or 1.

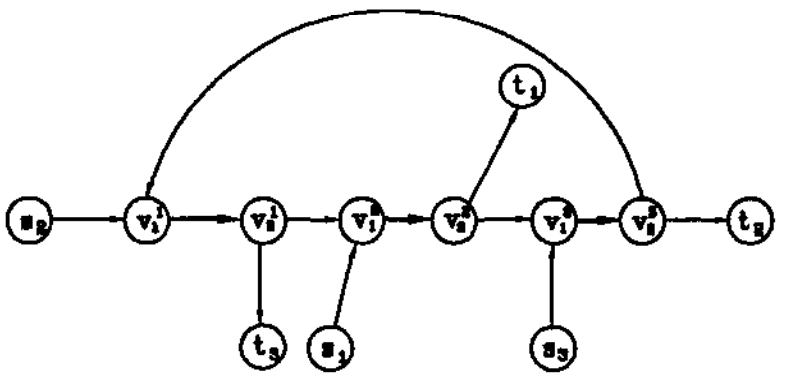
Fig. 7 shows two symmetric configurations (a) and (b). Configuration (b) is obtained from configuration (a) by cyclic permutation of source–destination nodes.

In Fig. 8, configuration (d) is obtained from configuration (c) by setting the number of shared segments of $s_3 - t_3$ to zero.

An optimal solution defined in this section actually satisfies the ladder property for any combination of two source–destination pairs except that the two shared segments between P_1 and P_2 in Fig. 5 are apparently shared in the same order. Such an apparent anomaly occurs because of the modification of the definition of the shared segment.



Configuration (a)

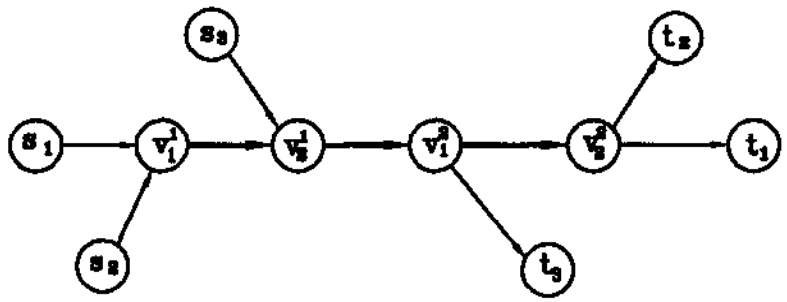


Configuration (b)

Fig. 7. An illustration of symmetric configurations.

However, the property is not really violated because the intermediate segment $v_2^1 \rightsquigarrow v_1^2$ has been shared by these paths with another path P_3.

To devise a dynamic programming algorithm, we would like to show that the problem can be decomposed into several subproblems with three intermediate nodes, say, a_1, a_2 and a_3. Before proceeding with the dynamic programming formulation, we study a specific example to explain this decomposition. Fig. 9 is an example network with three sources s_1, s_2, s_3, and their corresponding destinations t_1, t_2, t_3. By carefully examining all possible subgraphs of this network that connect our source–destination pairs, we find that the optimal solution of the point-to-point connection is as shown in Fig. 10 with a total cost of 34. Our aim is to look for a decomposition that will generate two arc-disjoint subgraphs. There may be several decompositions possible for a given solution subgraph. In Fig. 11 we show one such possible decomposition.



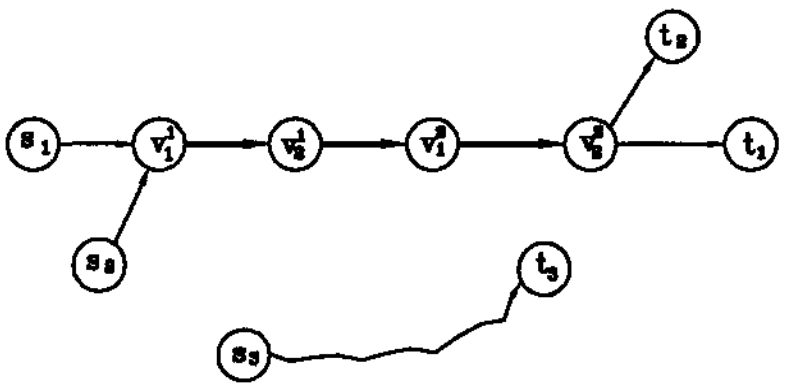


Fig. 8. An illustration of a degenerate configuration.

Two subgraphs are generated by a decomposition of the original solution graph along three nodes. Only these nodes are common between these subgraphs. In our specific example, two nodes are identical, i.e., $a_1 \equiv a_3 (\equiv v_1^2)$. Also, i_2 is chosen as a_2. The right subgraph in Fig. 11 has three shared segments (arcs). The segment $v_1^1 \rightarrow i_1$ is shared by paths P_1 and P_2. The segment $i_1 \rightarrow v_2^1$ is shared by all three paths; and the segment $v_2^1 \rightarrow v_1^2$ is shared between paths P_1 and P_3. Further, the left subgraph has two shared segments and can be easily identified. This example demonstrates how the following dynamic programming algorithm works.

3.1. *Dynamic programming algorithm*

We focus on devising a dynamic programming algorithm for $p = 3$, in this section. We start by presenting a lemma bounding the number of shared segments of an

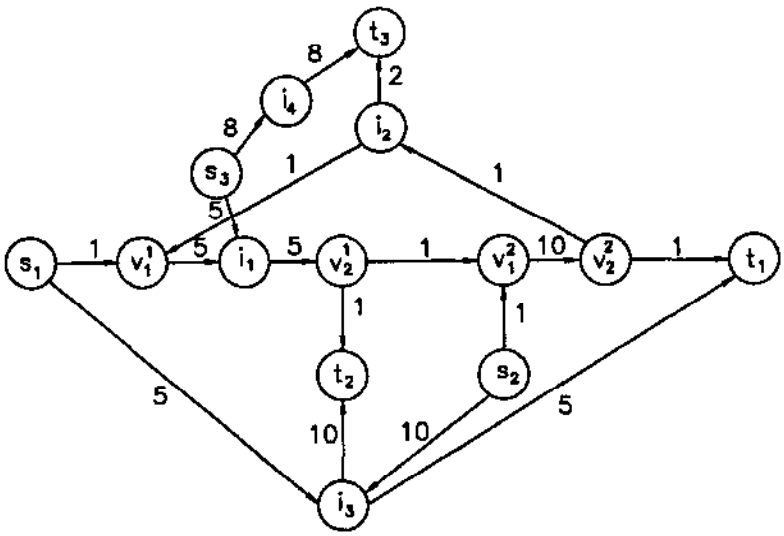


Fig. 9. An example network for $p = 3$.

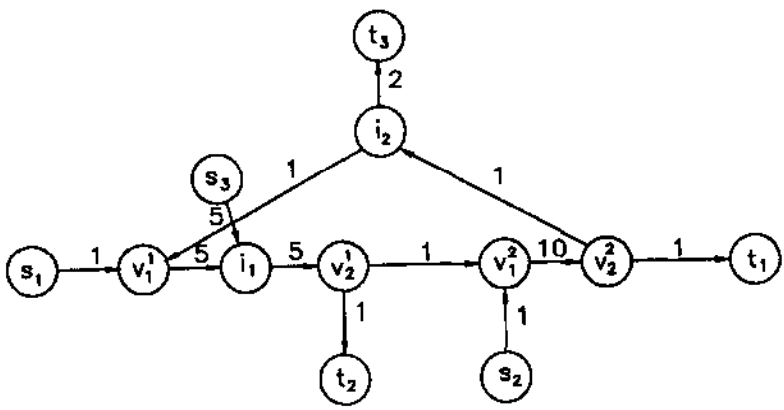


Fig. 10. An optimal solution for an example network.

optimal solution for $p \geqslant 3$, which in turn determines the number of iterations of the algorithm.

Lemma 4. *An upper bound on the number of shared segments in an optimal solution of a point-to-point connection problem with* $p \geqslant 3$ *is* $\lfloor (n-1)p/2 \rfloor$.

Proof. The number of shared segments in an optimal solution of a point-to-point connection problem cannot exceed the number of shared arcs. Since the number of

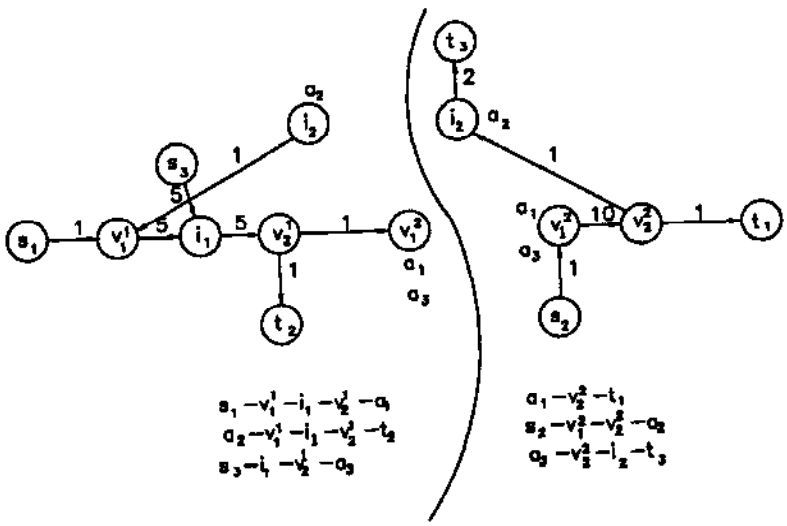


Fig. 11. An optimal decomposition.

arcs on each path is at most $(n-1)$, we have a total of $p(n-1)$ arcs on those p paths. Also, we need at least two paths to share a segment. Therefore, the number of shared segments cannot exceed $\lfloor (n-1)p/2 \rfloor$. □

Now, we describe a dynamic programming formulation for the problem with $p = 3$.

1. Define the optimal value function $\mathscr{F}(u_1, u_2, u_3; v_1, v_2, v_3; m) \equiv$ optimal value of the point-to-point connection problem in G with sources u_1, u_2, u_3, the corresponding destinations v_1, v_2, v_3, and with up to m shared segments among the paths connecting those source–destination pairs.
2. Define the recurrence relation: (for $m \geqslant 2$)

$$\mathscr{F}(u_1, u_2, u_3; v_1, v_2, v_3; m) \equiv \min \begin{cases} \mathscr{F}(u_1, u_2, u_3; v_1, v_2, v_3; m-1), \\ \mathscr{F}'(u_1, u_2, u_3; v_1, v_2, v_3; m), \end{cases}$$

where

$$\mathscr{F}'(u_1, u_2, u_3; v_1, v_2; v_3; m)$$

$$\equiv \min_{\substack{i \in \{1,2,\ldots,m-1\} \\ a_1, a_2, a_3 \in V}} \begin{cases} \mathscr{F}(u_1, a_2, a_3; a_1, v_2, v_3; m-i) + \mathscr{F}(a_1, u_2, u_3; v_1, a_2, a_3; i), \\ \mathscr{F}(a_1, u_2, a_3; v_1, a_2, v_3; m-i) + \mathscr{F}(u_1, a_2, u_3; a_1, v_2, a_3; i), \\ \mathscr{F}(a_1, a_2, u_3; v_1, v_2, a_3; m-i) + \mathscr{F}(u_1, u_2, a_3; a_1, a_2, v_3; i), \\ \mathscr{F}(u_1, u_2, u_3; a_1, a_2, a_3; m-i) + \mathscr{F}(a_1, a_2, a_3; v_1, v_2, v_3; i). \end{cases}$$

3. Specify the boundary value condition:

$$\mathscr{F}(u_1, u_2, u_3; v_1, v_2, v_3; 1) \equiv \min \begin{cases} \mathscr{F}_1(u_1, u_2, u_3; v_1, v_2, v_3; 0), \\ \mathscr{F}_1'(u_1, u_2, u_3; v_1, v_2, v_3; 1), \end{cases}$$

where

$$\mathscr{F}_1(u_1,u_2,u_3;v_1,v_2,v_3;0) = C^*(u_1,v_1) + C^*(u_2,v_2) + C^*(u_3,v_3)$$

and

$$\mathscr{F}_1'(u_1,u_2,u_3;v_1,v_2,v_3;1)$$

$$= \min_{a_1,a_2 \in V} \begin{cases} C^*(u_1,a_1) + C^*(u_2,a_1) + C^*(a_1,a_2) + C^*(a_2,v_2) \\ \quad + C^*(a_2,v_2) + C^*(u_3,v_3), \\ C^*(u_1,a_1) + C^*(u_3,a_1) + C^*(a_1,a_2) + C^*(a_2,v_1) \\ \quad + C^*(a_2,v_3) + C^*(u_2,v_2), \\ C^*(u_2,a_1) + C^*(u_3,a_1) + C^*(a_1,a_2) + C^*(a_2,v_2) \\ \quad + C^*(a_2,v_3) + C^*(u_1,v_1), \\ C^*(u_1,a_1) + C^*(u_2,a_1) + C^*(u_3,a_1) + C^*(a_1,a_2) \\ \quad + C^*(a_2,v_1) + C^*(a_2,v_2) + C^*(a_2,v_3). \end{cases}$$

4. Output the answer: $\mathscr{F}(s_1,s_2,s_3;t_1,t_2,t_3;K)$, where $K = \lfloor (n-1)p/2 \rfloor$.

The definition of the optimal value function is rather standard. We will explain the recurrence relation later in the proof of Theorem 6. As to the boundary condition, the first term is the sum of the costs of the shortest paths between the corresponding source–destination pairs because it does not have any shared segments. The second term is realized when we pick up any two source–destination pairs or all three pairs with a shared segment between them. Note that all costs in the calculation of this term must be from the shortest paths and the shared segment is only counted once. Figs. 12–15 represent their corresponding configurations.

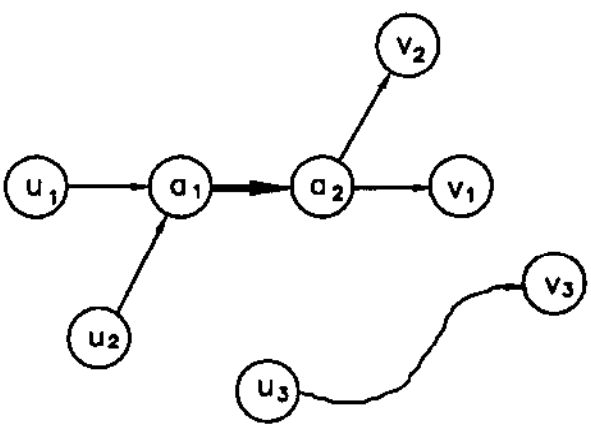


Fig. 12. Configuration 1.

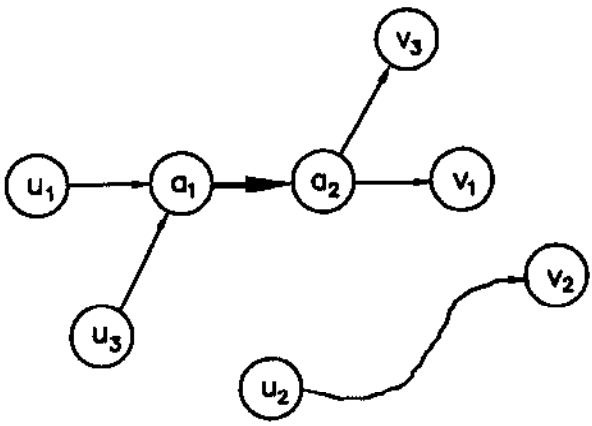


Fig. 13. Configuration 2.

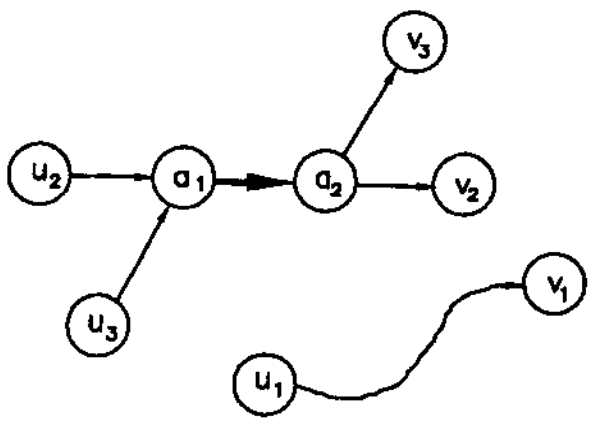


Fig. 14. Configuration 3.

Now, we have to prove that the decomposition in the above algorithm is possible. The proof is long due to the lack of any special structure for the optimal solution. To understand the proof of the following Optimal Decomposition theorem, a particular attention should be paid to symmetric configurations.

Theorem 5 (Optimal Decomposition Theorem). *Let $G(P)$ be the graph induced by any optimal solution $P = \{P_1, P_2, P_3\}$ consisting of the paths for the three source–destination pairs of the point-to-point connection problem. Then, there exist nodes $\{a_1, a_2, a_3\} \in G(P)$ such that $G(P)$ can be decomposed along those nodes into two arc-disjoint subgraphs $G^1(P)$ and $G^2(P)$ of the form shown in the recurrence relation. Moreover, these subgraphs denote the optimal solution of the point-to-point connection problem with their corresponding source–destination pairs.*

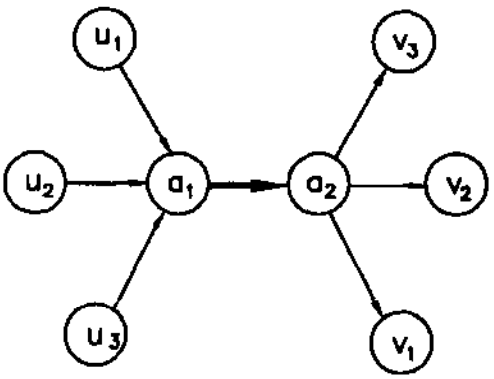


Fig. 15. Configuration 4.

Proof.

(Part 1) First we assume the existence of the decomposition nodes a_1, a_2, a_3, and prove that the subgraphs generated by the decomposition are optimal solutions of the corresponding point-to-point connection problems. Let us denote the total cost of the optimal solution with graph $G(P)$ by $|G(P)|$. The existence of nodes that decompose $G(P)$ into arc-disjoint subgraphs $G^1(P)$ and $G^2(P)$ implies $|G(P)| = |G^1(P)| + |G^2(P)|$. If there exists a subgraph $G'(P)$ with $|G'(P)| < |G^1(P)|$ between the same source–destination pairs as that of $G^1(P)$, then $G^1(P)$ could be replaced in $G(P)$ by $G'(P)$ to get $\tilde{G}(P)$. Hence, $|\tilde{G}(P)| = |G'(P)| + |G^2(P)| < |G^1(P)| + |G^2(P)| = |G(P)|$. But, this would contradict the hypothesis that $G(P)$ is the subgraph induced by the optimal solution. Thus, the claim follows.

(Part 2) We prove the existence of the decomposition nodes. Two cases based on the number of shared segments between the paths need to be discussed. Let x_{ij} denote the number of segments shared in the reverse order by path P_i and path P_j. Let x_{123} denote the number of segments shared by all three paths.

Case 1: $x_{ij} \leqslant 1, \forall i, j$. There are two subcases here. When $x_{123} > 0$, the two cases of Figs. 5 and 6 with their symmetric and degenerate configurations take place. In this situation, we can decompose the problem into two subproblems using just one node. When $x_{123} = 0$, the case in Fig. 16, with its symmetric and degenerate configurations occurs and choosing nodes a_1, a_2, a_3 as indicated in that figure we can decompose the problem into two arc-disjoint subgraphs.

Case 2: *Otherwise*. Here again we have two subcases as follows. Consider all the possible pairs from three paths that have segments shared in the reverse order. The two subcases result by realizing that either there exists at least one pair (without loss of generality, paths P_1 and P_2) with last (with respect to each path) shared segment or any part of it, not shared by the third path; or there does not exist such a pair.

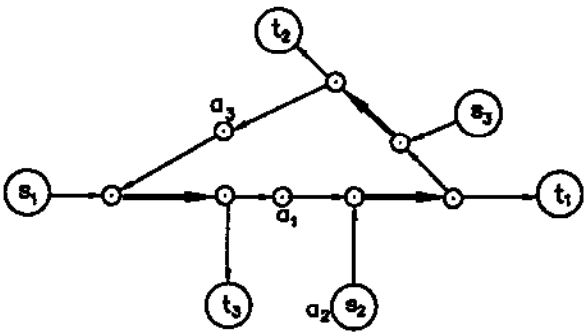


Fig. 16. Case 1.

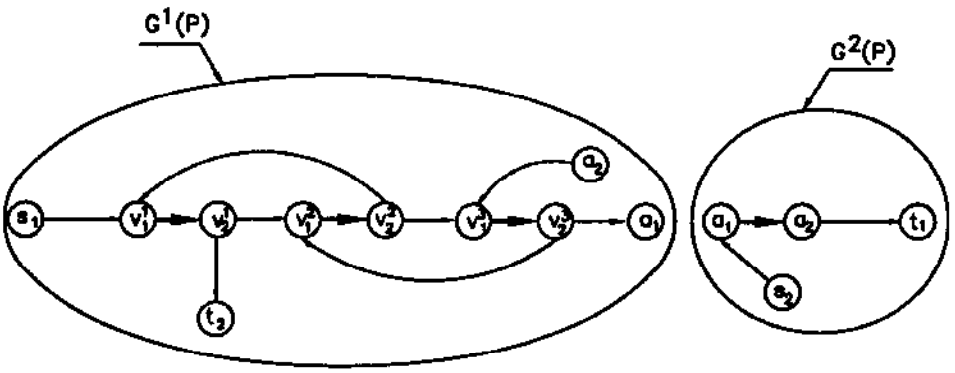


Fig. 17. Case 2(i).

(i) In this case, we assume there exists such a pair. Then, the end node of the segment can be chosen as a_2, while the start node is chosen as a_1. Now, consider the graph $G(P)$ induced by these two paths under consideration. It follows by construction that a_1 and a_2 will decompose this subgraph further into two subgraphs which do not share any arcs. Such a decomposition, with the disjoint subgraphs $G^1(P)$ and $G^2(P)$, is illustrated in Fig. 17.

The third decomposition node is chosen as the node where the third path leaves one of the subgraphs permanently after sharing some arcs in that subgraph. It is not difficult to see that the path cannot come back and share a segment with the old subgraph. If such a case existed, we can find a better solution by eliminating some of the arcs. But, this contradicts the hypothesis that we started with an optimal solution and hence, the subgraphs must be disjoint. To clarify the above argument, take a look at Fig. 17. Consider s_3–t_3 path first using arcs in $G^2(P)$ and then using arcs in $G^1(P)$. Now, it has to use either arcs in s_2–a_1 segment or in a_2–t_1 segment. If it uses arcs in the first segment, then it cannot use any arcs from a_2–t_2. Thus, it can only use some

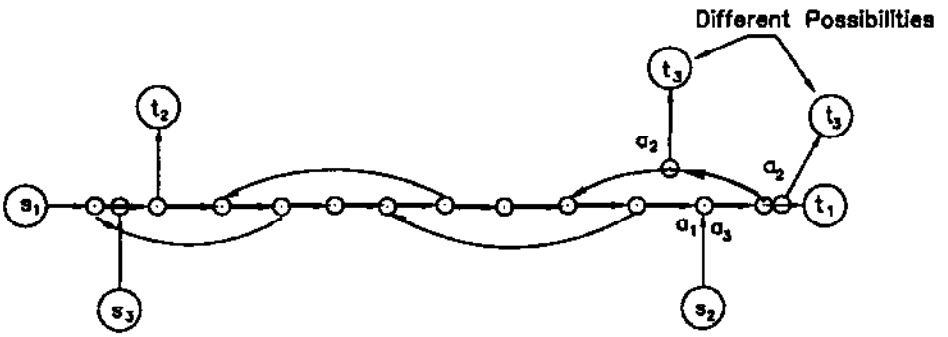


Fig. 18. Case 2(ii).

initial arcs of the s_1 a_1 path in $G^1(P)$. But, if indeed that is the situation then we can show that the s_3 t_3 path can neither use the a_2–t_1 segment nor the s_2 a_1 segment in $G^2(P)$. The first case is obvious, while if the path uses some initial arcs of the s_2–a_1 segment, we can find an alternate route for the s_1–t_1 path to improve the solution. This causes a contradiction. Similarly, we can prove the other case involving the use of the a_2–t_1 segment. For the initial use of subgraph $G^1(P)$, the case follows with the same reasoning. This proves that our choice of the third node achieves the desired decomposition into two disjoint subgraphs.

(ii) In the second case, we assume that there does not exist a pair as described earlier. The only possible situation is shown in Fig. 18.

In Fig. 18, we can choose a_1 as before, a_2 as the node where the paths P_2 and P_3 depart from each other, and a_3 the same as a_1. The analysis is similar to the earlier subcase and hence, the decomposition into two disjoint subgraphs is accomplished. This proves the existence of the decomposition nodes. □

Using Theorem 5, we now show that the proposed dynamic programming algorithm generates a correct answer in $\mathcal{O}(n^{11})$ time.

Theorem 6. *The dynamic programming formulation solves the point-to-point connection problem with three source–destination pairs on directed networks in $\mathcal{O}(n^{11})$ time.*

Proof. From Theorem 5, it follows that we can decompose the problem into smaller subproblems. The recurrence relation considers all possible 2^{p-1} cases resulting from the decomposition along any choice of three nodes. Note that for $p = 3$, there are four cases.

Preprocessing to compute all $C^*(u, v)$ takes $\mathcal{O}(mn + n^2 \log n)$ time as mentioned in the proof of Theorem 3. We also have to solve the recurrence relation for all possible combinations of sources, destinations, and the number of shared segments. For $p = 3$, the total number of all possible combinations in $\mathcal{O}(n^7)$. If we use the same amount of storage to store all intermediate results, and since each evaluation of the recurrence relation takes $\mathcal{O}(n^4)$ time, the overall complexity is $\mathcal{O}(n^{11})$. □

4. A conjecture for $p > 3$

We conjecture that the dynamic programming formulation discussed in Section 3.1 can be extended for $p > 3$. In this section, we sketch a recurrence relation and discuss the potential difficulties in providing a complete proof. First, let us define that

$$S = \{u_1, u_2, \ldots, u_p\},$$

$$T = \{v_1, v_2, \ldots, v_p\},$$

$$S^* = \{s_1, s_2, \ldots, s_p\},$$

$$T^* = \{t_1, t_2, \ldots, t_p\},$$

$$X \subseteq \{2, 3, \ldots, p\} = N,$$

$$S_1 = \{u_1\} \cup \{u_i \mid i \in X\} \cup \{a_i \mid i \notin X, i \in N\},$$

$$T_1 = \{a_1\} \cup \{a_i \mid i \in X\} \cup \{v_i \mid i \notin X, i \in N\},$$

$$S_2 = \{a_1\} \cup \{u_i \mid i \notin X, i \in N\} \cup \{a_i \mid i \in X\},$$

$$T_2 = \{v_1\} \cup \{a_i \mid i \notin X, i \in N\} \cup \{v_i \mid i \in X\}.$$

A dynamic programming formulation is proposed as follows:

1. Define the optimal value function $\mathscr{F}(S, T, m) \equiv$ Optimal value of the point-to-point connection problem in G with sources $u_1, u_2, \ldots, u_p$ and the corresponding destinations $v_1, v_2, \ldots, v_p$ and with upto m shared segments between the paths connecting those source–destination pairs.
2. Define the recurrence relation:

$$\mathscr{F}(S, T, m) \equiv \min \begin{cases} \mathscr{F}(S, T, m-1), \\ \mathscr{F}'(S, T, m), \end{cases}$$

where

$$\mathscr{F}'(S, T, m) \equiv \min_{\substack{i \in \{1, 2, \ldots, m-1\} \\ a_1, a_2, \ldots, a_p \in V \\ \forall X}} \{\mathscr{F}(S_1, T_1, m-i) + \mathscr{F}(S_2, T_2, i)\}.$$

3. Specify the boundary value condition:

$$\mathscr{F}(S, T, 1) \equiv \min \begin{cases} \mathscr{F}(S, T, 0), \\ \mathscr{F}'(S, T, 1), \end{cases}$$

where

$$\mathscr{F}(S, T, 0) = C^*(u_1, v_1) + C^*(u_2, v_2) + \cdots + C^*(u_p, v_p).$$

Let $Y \subseteq \{1, 2, \ldots, p\}$ with $2 \leqslant |Y| \leqslant p$, then,

$$\mathscr{F}'(S, T, 1) = \min_{\substack{Y \\ a_1, a_2 \in V}} \begin{cases} C^*(a_1, a_2) + \sum_{i_k \in Y} \{C^*(u_{i_k}, a_1) + C^*(a_2, v_{i_k})\} \\ + \sum_{i_j \notin Y} C^*(u_{i_j}, v_{i_j}). \end{cases}$$

4. Output the answer: $\mathscr{F}(S^*, T^*, K)$ where $K = \lfloor (n-1)p/2 \rfloor$.

Conjecture 7. *The dynamic program discussed above solves the point-to-point connection problem on directed graphs with p (fixed) source–destination pairs in $\mathcal{O}(n^{3p+2})$ time for $p > 3$.* □

It is not difficult to prove that the complexity of this algorithm is $\mathcal{O}(n^{3p+2})$. The difficulty is with the proof of correctness of such a decomposition scheme. Some major difficulties we encountered in proving the existence of such a decomposition scheme include:

1. The constructive proof as shown in Section 3.1 is not easy to generalize because of the highly complex structure of an optimal solution.
2. Our conjecture for $p > 3$ uses a decomposition scheme along a set of nodes of the graph. Therefore, a complete proof for the conjecture using induction on the number of source destination pairs does not work.

5. Remarks

1. Our implementation for $p = 2$ is more efficient than the algorithm proposed by Li et al. [10].
2. Clearly, our algorithm for $p = 3$ can be specialized for $p = 2$. It can also be reduced to a correct version of Li et al. [10] by making use of symmetry. Also, without any insight into the structure of an optimal solution, dynamic programming leads to state space explosion. Hence, the algorithm is inefficient. In this case, the development of an efficient algorithm will depend on identifying structural properties of an optimal solution.
3. We have a conjecture to solve the problem for $p > 3$. A complete proof or a counterexample to show the impossibility of our decomposition scheme will be interesting. Of course, the problem may need a totally different approach. Again, the algorithm subjects to *curse of dimensionality* and leads to high complexity.
4. There exist polynomial time heuristics with guaranteed good performance for the point-to-point connection problem on undirected networks. However, it is not apparent to us how to redesign these good heuristics for directed networks.

Acknowledgements

We would like to thank Dr. Mark Hartman of the University of North Carolina, Chapel Hill, for pointing out Ref. [1] to us, and for suggesting an efficient

implementation of our algorithm for $p = 2$. Also, the first author would like to thank Dr. Matt Stallmann and Dr. Rex Dwyer for a useful discussion about the problem.

References

[1] M.O. Ball, W.-G. Liu and W.R. Pulleyblank, Two-terminal Steiner tree polyhedra, in: Contributions to Operations Research and Economics (MIT Press, Cambridge, MA, 1989) 251–284.
[2] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, (North-Holland, New York, 1976).
[3] N. Christofides and C.A. Whitlock, Network synthesis with connectivity constraints a survey, in: J.P. Brans, ed., Operational Research '81, (North Holland, Amsterdam, 1981).
[4] S.E. Dreyfus and R.A. Wagner. The Steiner problem in graphs, Networks 1 (1972) 195–207.
[5] M.L. Fredman and R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. ACM 34 (1987) 596 615.
[6] M.X. Goemans and D.P. Williamson, A general approximation technique for constrained forest problem, SIAM J. Comput., to appear.
[7] S.L. Hakimi, Steiner's problems in graphs and its applications, Networks 1 (1971) 113–133.
[8] F.K. Hwang, D.S. Richards and P. Winter, The Steiner Tree Problem (Elsevier, Amsterdam, 1992).
[9] J.M.Y. Leung, T.L. Magnanti and V. Singhal, Routing in point-to-point delivery systems: formulation and solution heuristics, Transportation Sci. 24 (1991) 245–260.
[10] C.L. Li, S.T. McCormick and D. Simchi-Levi, The point-to-point delivery and connection problems: complexity and algorithms, Discrete Appl. Math. 36 (1992) 267–292.
[11] T.L. Magnanti and R.T. Wong, Network design and transportation planning: models and algorithms, Transportation Sci. 18 (1984) 1–55.
[12] S. Martello and P. Toth, Finding a minimum equivalent graph of a digraph, Networks 12 (1982) 89–100.
[13] C.L. Monma, D.F. Shallcross, Methods for designing communication networks with certain two-connected survivability constraints, Oper. Res. 37 (1989) 531–541.
[14] M.G. Natu, S.-C. Fang, On the point-to-point connection problem, Inform. Process. Lett. 53 (1995) 333–336.
[15] M.B. Ritchey, R.G. Parker, R.L. Rardin, An efficiently solvable case of the minimum weight equivalent subgraph problem, Networks 15 (1985) 217–228.